Island Transpeciation: A Co-Evolutionary Neural Architecture Search, applied to country-scale air-quality forecasting

Konstantinos Theodorakos, Oscar Mauricio Agudelo, Joachim Schreurs,
Johan A.K. Suykens, *Fellow Member, IEEE* and Bart De Moor, *Fellow Member, IEEE & SIAM*

***Abstract*—Air pollution causes around 400.000 premature deaths per year in Europe due to Particulate Matter, nitrogen oxides (NO_x) and ground-level ozone (O_3) pollutants. Multiple-Input Multiple-Output Nonlinear Auto-Regressive eXogenous Deep Neural Networks are frequently used to predict a day before, air-quality pollution incidents, at a country-scale. With complexity and data sizes increasing, finding optimal models becomes harder. We propose "Island transpeciation" to optimize hyperparameters and architectures. Unlike single "off-the-shelf" optimizers, island transpeciation combines results from multiple optimizers, to consistently provide excellent performance. Moreover, we show that island transpeciation outperforms random model search and other previous modelling efforts. Island transpeciation is a Neural Architecture Search (NAS) that uses co-evolution (genes), to combine (transpeciation) populations of incompatible optimizers (species). In island transpeciation, heterogeneous hardware resources can be parallelized with fault tolerance and distributed control. We have successfully used these techniques to predict next-day O_3 concentrations across the Belgian territory.**

***Index Terms*—deep neural networks, neural architecture search, co-evolution, air quality forecasting**

I. Introduction

AIR pollutants are substances that harm the human health and the environment. They can be released from natural and anthropogenic sources: mining activities, agriculture, waste treatment, industrial processes, energy plants, burning fossil fuels, road transportation, residential activities and natural phenomena. Total air pollution was the 5th global risk factor in 2017 [1], by total number of deaths from all causes, ages and both sexes. Air pollution is the cause of around 400.000 premature deaths per year and is the one of the most significant health risks in Europe [2]. In addition, there is a negative economic impact by reducing productivity through working days lost, increasing health care costs and human life span shortening. The most dangerous pollutants in Europe are Particulate Matter (PM), Nitrogen Oxides (NO_x) and ground-level ozone (O_3).

Corresponding author: Konstantinos Theodorakos. Preprint submitted on April 12, 2021.

The authors are with KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium (e-mails: {konstantinos.theodorakos, mauricio.agudelo, joachim.schreurs, johan.suykens, bart.demoor}@esat.kuleuven.be).

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

Air pollution forecasting is a prevention measure against the short-term (hours or days) and long-term (years) exposure to harmful substances for vegetation, animals and humans. With accurate forecasting, governments and policy-makers could raise real-time "low air-quality" alerts. With timely warnings, the public minimizes negative health effects by reducing outdoor activities during dangerous times.

Multiple-Input Multiple-Output (MIMO), Nonlinear Auto Regressive eXogenous (NARX) Deep Neural Networks (DNN) for air-quality forecasting is an "all-in-one" modelling architecture that can predict next-day O_3 concentrations, at a country scale. In this work, we used real-world, publicly available data: Time-series (1990 to 2018) from 46 O_3 Belgian monitoring stations retrieved from the European Environmental Agency (EEA) [3], which we combined with 51 weather variables [4] acquired from the surface-level ERA-interim European Centre for Medium-Range Weather Forecasts (ECMWF) database. The proposed DNNs successfully predicted one day before, an "inform-public" O_3 alert level in Belgium at critical times.

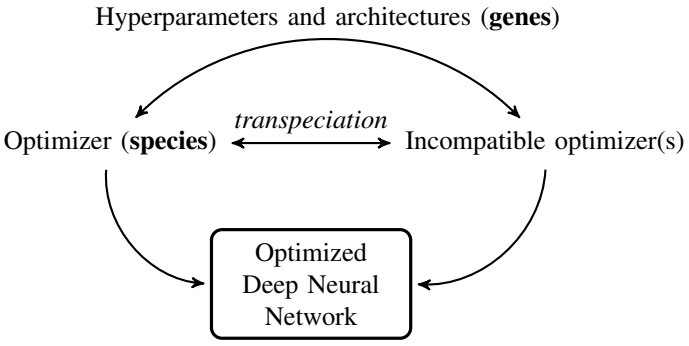


Figure 1. Island transpeciation is a global search technique that uses co-evolution (genes) to combine (transpeciation) populations of incompatible optimizers (species), in order to find optimal Deep Neural Network (DNN) models.

With complexity and data sizes increasing, finding optimal models becomes harder. To improve the forecasting performance of DNNs, we developed *island transpeciation* [5] (Fig. 1). Unlike "off-the-shelf" optimizers, island transpeciation can combine results from multiple optimizers, to consistently provide good performance. Island transpeciation is a Neural Architecture Search (NAS) technique that uses co-evolution

(genes) to combine (transpeciation) populations of incompatible optimizers (species), in order to find optimal DNN models.

Contributions:

- MIMO NARX DNN: A prototype for country-scale air quality forecasting, using a single model.
- Definition of a new operator for evolutionary algorithms: *transpeciation*.
- A new type of automated parallel and distributed NAS: Island transpeciation. Heterogeneous networked computing resources can cooperate with "hot-plugging" and fault-tolerance.
- Deep learning model configuration suggestions in the context of ozone forecasting.

This document is divided into 5 sections. Section II provides a brief introduction to DNNs, Neural Architecture Search and background information on air-quality standards and existing forecasting methods. Section III discusses the island transpeciation neural architecture search. Section IV describes the details of implementing island transpeciation and the MIMO-NARX DNN model prototype. Section V shows the performance of the air quality forecasting models and the effectiveness of island transpeciation. Section VI, contains insights and conclusions, Appendix A the data/code repositories and information on the CPU and Multi-GPU parallelism implementation. Appendix B presents the exogenous variables used in the models.

II. Background

A. Hyperparameter optimization of neural networks

Recurrent Neural Networks (RNN) [6] is a type of Artificial Neural Networks (ANN) that is commonly used to model time series. These RNNs exhibit temporal sequential behavior where memories are retained and recalled as stable entities collectively. An extension of RNN is the *Bidirectional* Recurrent Neural Networks (BRNN) [7], which use sequential data in both forward/backward time directions without the requirement of the input data length to be fixed. *Long Short-Term Memory (LSTM)* [8] cells store information over extended time intervals, without losing shorter time predictive capabilities. *Gated Recurrent Units (GRU)* [9] are similar to the LSTM, with less trainable parameters. LSTMs with a forget gate (Fig. 2), are expressed formally [8] as:

$$\begin{aligned}
\mathbf{f_t} &= \sigma_g(\mathbf{W_f x_t} + \mathbf{U_f h_{t-1}} + \mathbf{b_f}) \\
\mathbf{i_t} &= \sigma_g(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + \mathbf{b_i}) \\
\mathbf{o_t} &= \sigma_g(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + \mathbf{b_o}) \\
\mathbf{c_t} &= \mathbf{f_t} \circ \mathbf{c_{t-1}} + \mathbf{i_t} \circ \sigma_c(\mathbf{W_c x_t} + \mathbf{U_c h_{t-1}} + \mathbf{b_c}) \\
\mathbf{h_t} &= \mathbf{o_t} \circ \sigma_h(\mathbf{c_t})
\end{aligned} \tag{1}$$

where d is the number of input features, h is the number of hidden units, $\circ$ the Hadamard product. $\mathbf{x_t} \in \mathbb{R}^d$ is the input vector to the LSTM unit at time t, $\mathbf{f_t} \in \mathbb{R}^h$ is the forget gate's activation vector, $\mathbf{i_t} \in \mathbb{R}^h$ is the input gate activation vector, $\mathbf{o_t} \in \mathbb{R}^h$ is the output gate activation vector, $\mathbf{c_t} \in \mathbb{R}^h$ is the cell state vector and $\mathbf{h_t} \in \mathbb{R}^h$ is the hidden state vector. The $\mathbf{W} \in \mathbb{R}^{h \times d}$, $\mathbf{U} \in \mathbb{R}^{h \times h}$ weight matrices and the bias vectors

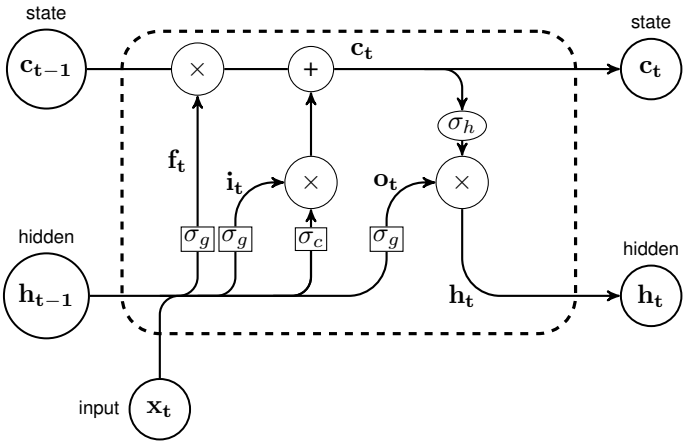


Figure 2. Diagram of a Long-Short Term Memory (LSTM) cell with a forget gate [8].

$\mathbf{b} \in \mathbb{R}^h$, are learned during training. σ_g is a sigmoid function and σ_c, σ_h are hyperbolic tangent functions.

LSTMs fall under a more general class of *Nonlinear Auto-Regressive eXogenous (NARX)* models [10]. NARX is a predictive mathematical formulation with output variables depending on previous values as well as exogenous variables, along with a stochastic (random) term:

$$\mathbf{y}(t) = \mathbf{F}(\mathbf{y}(t-1), ..., \mathbf{y}(t-l_y), \mathbf{u}(t), ..., \mathbf{u}(t-l_u)) + \mathbf{e}(t) \tag{2}$$

where $\mathbf{y}(t) \in \mathbb{R}^n$ is the output vector at time t (e.g., O_3 concentrations), $\mathbf{u}(t) \in \mathbb{R}^m$ is the vector of exogenous input variables (e.g., temperature, total cloud cover, etc.). l_y and l_u are the numbers of lags for $\mathbf{y}(t)$ and $\mathbf{u}(t)$ respectively. $\mathbf{e}(t) \in \mathbb{R}^n$ is the prediction error (white noise process assumed) and $\mathbf{F}(.)$ is a nonlinear vector function (ANN, polynomial function, etc). In this work, we focus on training *Multiple-Input Multiple-Output (MIMO)* models. These allow us to use multiple input time-series of air quality monitoring stations, to predict the outcomes of multiple stations. With *Multiple-Input Single-Output (MISO)*, we forecast the output of a single measuring station.

Due to the inherent difficulty of modern day prediction problems, such as the air quality forecasting that we focus on this paper, neural network type models typically have an abundance of hyperparameters to tune. We here give an overview of commonly used methods to explore and optimize these parameters: *Particle Swarm Optimization* (PSO) [11] solves problems iteratively, using populations of moving candidate solutions (particles), located within a parameter space. Particles (swarm) influence each-other based on their fitness, position, velocity etc. *Genetic Algorithms* (GA) [12] is a class of Evolutionary Algorithms (EA) [13] that generate candidate solutions via natural selection and biology-inspired operators: mutation, crossover and selection. *Differential Evolution* (DE) [14] is an EA that solves non-differentiable optimization problems, by combining multiple elite candidate solutions. *Bayesian Optimization* (BO) [15] defines prior and posterior distributions (Gaussian process) over objective functions, using exploration/exploitation trade-offs on bounded parameter

spaces. *Random Search* (RS) [16] samples from random distributions to solve black-box optimization problems.

Neural Architecture Search (NAS) automatically architects Neural Network designs [17]. Due to the inherent complexity of DNNs and the tasks at hand, empirically architecting DNN can be tedious, time-consuming and ineffective. NAS is overlapping with the fields of meta-learning [18] and hyperparameter optimization and has three main components: (1) *Search space*: The bounded or unbounded range of architecture parameters. It can be prone to human bias. Careful treatment can significantly reduce the total NAS search space and execution time [19]. (2) *Search strategy*: The method to perform exploration and exploitation of the architecture search space. (3) *Performance estimation strategy*: The metric that can foresee the future performance of the trained ANN on unseen data and future tasks. *Neuroevolution* [20] is a type of NAS that uses EA to generate ANN rules, parameters, topologies [21]. ANN neuron weights, hyperparameters and topologies can be improved via the processes of evolution: reproduction, mutation, recombination, etc. Two main variants exist. (1) Direct encoding: It generates compact and fine-tuned architectures that require a lengthy training process. (2) Indirect encoding: Coarser but faster than direct encoding. Suited for specific use-cases. To find optimal deep learning models for the air-quality time-series forecasting, we used indirect encoding. Architecture search is performed in conjunction with hyperparameter tuning.

DNN models can be characterized by tenths up to hundreds of hyper-parameters and architectures, with varying levels of sensitivity. Manually tweaking the available options does not guarantee optimal model performance [22]. There is evidence that combining multiple optimization algorithms leads to improvements in global model performance compared to a homogeneous/single-optimizer case [23]. In this paper, we argue that it is optimal for fast convergence to combine different types of optimizers for NAS as well. The introduction of island transpeciation in NAS, allows to combine DNN architectures from vastly different global optimizers. This ensures consistently good performance over different initializations, parameter spaces or difficulty of the problem.

B. Air-quality standards

Ozone (O_3) is an inorganic molecule, a less stable allotrope of oxygen. It has industrial and consumer applications as an oxidant, but damages the mucous and respiratory tissues of humans and animals [24]. Exposure to O_3 with a concentration of 1 part per million (ppm) can affect the respiratory system. Exposure to 15 to 20 ppm, for 2 or more hours can be life-threatening. For the long term protection of human health and vegetation, specific O_3 concentration thresholds were set by the European Union [25]:

1) Background: 60 $\mu g/m^3$.
2) Healthy limit: 120 $\mu g/m^3$.
3) Public informing (>1 station): 180 $\mu g/m^3$.
4) Alert: 240 $\mu g/m^3$.

We used real-world, publicly available data: Time-series (1990 to 2018) from 46 O_3 Belgian monitoring stations retrieved from the European Environmental Agency (EEA) [3]. Data are sampled as a maximum daily 8-hour mean by the following monitoring station types:

1) Urban: Protection of human health. Range: a few km^2. Location: residential areas of cities.
2) Suburban: Protection of human health and vegetation. Range: tens of km^2. Location: city outskirts.
3) Rural: Protection of human health and vegetation. Range: sub-regional levels (a few km^2). Location: small settlements, crops, forests and natural ecosystems.
4) Background-rural: Protection of vegetation and human health. Range: regional, national or continental (1.000 up to 10.000 km^2). Location: crops, forests, natural ecosystems, very low population regions.

C. Existing air-quality forecasting methods

In [26], the authors forecast daily PM in Belgium, using basic Artificial Neural Networks. A single model per monitoring station was proposed, with only a few exogenous variables. As a consequence, the numerical accuracy of the predicted concentrations was limited. The authors identified cloud cover, day of the week and wind direction as important exogenous features for forecasting, which we also include in our proposed model. Ensemble Kalman Filters (EnKF) [27] and Optimal Interpolation (OI) [28], improved the O_3 and PM estimates of the air-quality model AURORA in Belgium. Although clustering cannot offer direct numerical predictions for air-quality stations, it is worth mentioning. Incremental Kernel Spectral Clustering (IKSC) [29] clustered drifting non-stationary time-series of multiple PM monitoring stations over Belgium and three surrounding countries. Their approach captured some spatial shifting patterns on the spread of a pollution episode. In [30], the authors used Deep Learning to forecast O_3. The exogenous variables were reduced using a decision classification tree. A human-in-the-loop approach was used to tune a few DNN hyperparameters manually and to avoid overfitting. As exogenous parameters, other than meteorological variables the authors added chemical analytes. To our knowledge, chemical concentration forecasts are not easily obtainable and may require the additional dependency of the outputs of (computationally demanding) deterministic air-quality models. The authors used only 2 years of training data, with 80% training, 20% testing split, without any form of cross-validation or automated architecture search. Finally, this approach seems to work only for single monitoring station predictions.

III. Main result: Island Transpeciation

Overview

Speciation is the evolutionary process with which populations evolve to become distinct species [31]. Island transpeciation is a global search technique that uses evolution to combine incompatible optimizers, in order to find optimal DNN architectures (Fig. 3). Artificial populations of optimizers can be combined into panmictic structures of complex system networks called *island or multi-population models* [32]. Every optimizer maintains a number of candidate solutions in their

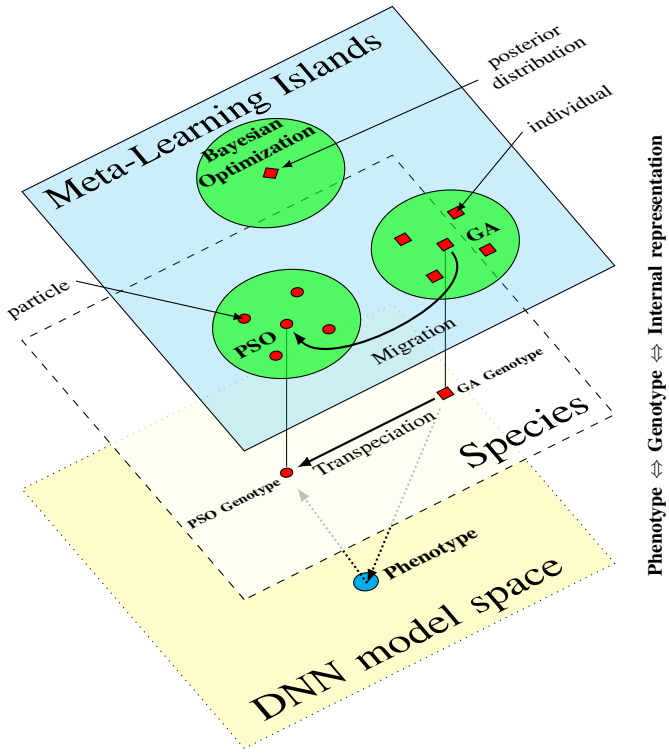


Figure 3. Island transpeciation is a co-evolutionary Neural Architecture Search (NAS) that optimizes DNN models (phenotype). Optimizers (top layer: meta-learning islands) like Bayesian Optimization, Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) can co-evolve DNN models (bottom layer: DNN model space). Cooperation and competition between incompatible optimizers is achieved via migration (transpeciation operator) of solutions between optimizers (mid layer: species) using: *genotype-to-internal representation* transformations.

internal representation level, in an *island*. In Fig. 3 we have three islands of the following iterative optimizers: Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Bayesian Optimization (BO). PSO and GA islands have a population of five (internal candidate solutions), whereas BO only one (distribution of parameters). An optimizer that can not directly use candidate solutions from other islands, is considered a unique *species*. E.g., a PSO algorithm (which contains individuals of the species "particle") can not directly accept and utilize candidate solutions from the "individual" species of a GA.

Transpeciation Operator

Transpeciation is an evolutionary operator that allows *internal representation* to (globally compatible) *genotype* transformations. In other words, to enable transferring of candidate solutions between incompatible optimizers, we project solutions from the internal algorithm format to a commonly compatible form. *This way, we can achieve concurrent cooperation between vastly different optimization algorithms.* The *transpeciation operator* is optimizer/implementation-dependent. For example (Fig. 3), transpeciation can transform an individual (from the GA species) into a particle (to the PSO species) and allow GA and PSO to co-operate seamlessly. An individual GA solution can migrate to a PSO island, after it is transformed (transpeciated) into the particle species via a common *genotype*. The bottom layer represents the final form of an optimized and trained DNN model, expressed as *phenotype*.

A. Generalized Island Model formal extension

From the Generalized Island Model (GIM) [23], we have *Archipelago* $\mathbb{A}$ which is the tuple:

$$\mathbb{A} =< \mathbb{I}, \mathscr{T} > \tag{3}$$

where i is the island identification with $i = 1, 2, ...n$, $\mathbb{I}$ the set of islands $I_i = \{I_1, I_2, ..., I_n\}$ and $\mathscr{T}$ the migration topology. Every island I_i is a tuple:

$$I_i =< \mathscr{A}_i, P_i, \mathscr{P}_i, \mathscr{R}_i > \tag{4}$$

where $\mathscr{A}_i$ is the optimization algorithm, P_i is the candidate solution population, μ_i is the migration interval, $\mathscr{P}_i$ is the migration-selection policy and $\mathscr{R}_i$ is the migration-replacement policy of island i.

Our extension on the original formalism is the transpeciation operator $\mathbb{T}$. Lines 4, 7 of Algorithm 1 extend GIM [23] (with μ as the migration interval). The *deme* $\mathbb{M}$ is a globally compatible sub-population, generated by the island i. $\mathbb{M}$ is created by the *internal representation* population of the island i, using the transpeciation operator $\mathbb{T}_i$:

$$\mathbb{M} = \mathbb{T}_i(internal_representation_i) \tag{5}$$

where $\mathbb{T}_i$ is the *transpeciation operator* for island i, $\mathbb{M}$ is a *deme*, a sub-population of globally compatible candidate solutions to be sent to adjacent islands. For example, the internal representation of a PSO algorithm has a "particle species" and for a BO a "distribution species". Typically, we would only be able to combine candidates from the same species, with the same number and type of arguments. However, transpeciation allows the back and forth cooperation of these incompatible optimizers.

The operator $\mathbb{T}'$ performs the reverse (sometimes lossy) transpeciation operation. Island i converts the globally compatible deme $\mathbb{M}'$ that was received from the adjacent islands, into the $internal_representation_i$:

$$internal_representation_i = \mathbb{T}'_i(\mathbb{M}') \tag{6}$$

where $\mathbb{M}'$ is the *deme* received from islands adjacent to I_i and $\mathbb{T}'_i$ is the *Transpeciation' operator* of island i, for reverse transformations (global to internal representations). A lossy case is when we transpeciate a particle from the PSO species into an individual of the DE species: we retain and convert all the particle position parameters. However, we lose the internal parameters regarding particle velocity or acceleration. For an optimization algorithm like random search there is no inverse transpeciation operation, because this algorithm does not accept demes internally.

In practice, the transpeciation operator performs mainly rescaling on hyperparameter bounds, from the internal representation format of each optimizer (i.e., for random search $\in [0, 1]$) to the globally compatible genotype bounds (i.e., for

Algorithm 1: ISLAND I_i WITH TRANSPECIATION OPERATOR $\mathbb{T}_i$

initialize P
while *NOT(stop criteria)* **do**
  $P' \leftarrow \mathscr{A}_i(P, \mu_i)$
  $\mathbb{M} \leftarrow \mathbb{T}_i(\mathscr{P}_i(P'))$ `// Transpeciation`
  Send $\mathbb{M}$ to islands adjacent to $I_i \in \mathscr{T}$
  Let $\mathbb{M}'$ be the set of solutions received from adjacent islands
  $P'' \leftarrow \mathscr{R}(P', \mathbb{T}'_i(\mathbb{M}'))$ `// Transpeciation'`
  $P \leftarrow P''$
end

the "unit" hyperparameter $\in [64, 512]$]). The *inverse transpeciation* operator performs the opposite transformation. Special cases may require additional operations, like with the Bayesian optimizer [15], which has a distribution as internal representation. The transpeciation operation in that case, additionally performs a "maximize" function call, where we get the Maximum A-Posteriori (MAP) hyperparameter estimation. In other words, we get the mode of the posterior parameter distribution, which is the "best suggested" hyperparameter point sample. For the inverse transpeciation operation, we additionally have a "probe" function call, where we "guide/suggest" solutions (coming from the other islands) to the Bayesian optimizer.

Algorithmic flow

Fig. 4 shows a flow chart of the island transpeciation NAS. There are three main phases: *global optimization*, *transpeciation* and *local optimization*. In the global optimization phase, every island is searching for DNN hyperparameters using a common genotype. Each internal model candidate that an optimizer species generates, is trained using a distributed worker (hardware resources) and evaluated for fitness. Periodically, migrations send and receive candidate models to/from the other adjacent islands. Out of all the neighboring models received, one is chosen probabilistically, using rank selection. The selected candidate is transformed into the internal representation of the island via transpeciation. E.g., for a PSO [11] island, the genotype received is transformed into a particle, using the genes as location and an arbitrary velocity. The accepted model can now replace the worst internal model of the island since it has the same species. In the case of a PSO island, the particle with the worst Mean Squared Error (MSE) is replaced. Island transpeciation allows for diverse global search methods to co-evolve models concurrently.

With island speciation, neuroevolution can be parallelized: Each island [32], [33] could run on a distinct computing thread, process or even a different computing node. Even though the evolution progresses distinctly in each island, populations of meta-learning individuals could migrate periodically between islands. Table I shows the numerical non-differentiable bounded global and local optimizers that we used as meta-learning islands.

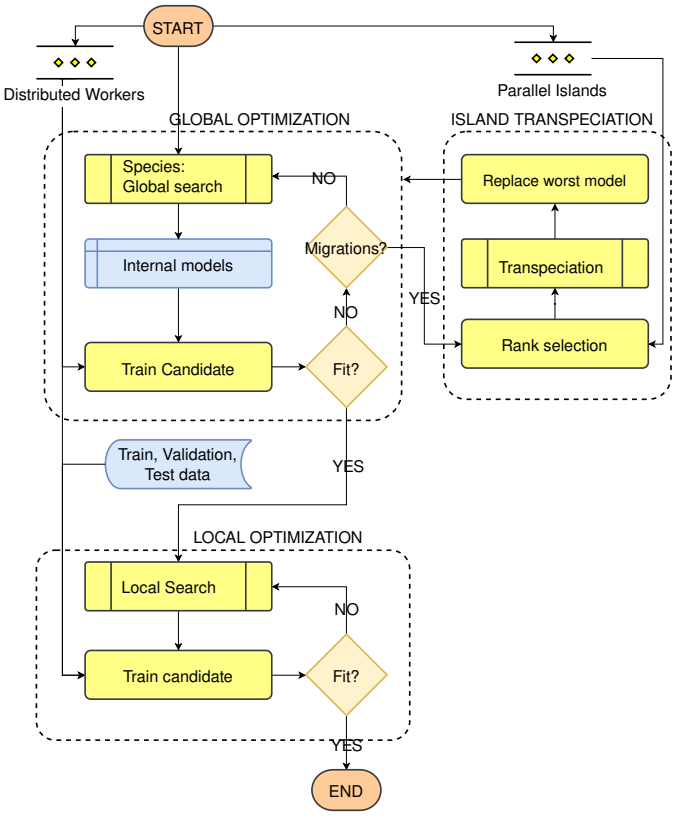


Figure 4. Flow chart of island transpeciation. *Global optimization* allows diverse global search methods to co-evolve DNNs in parallel via *transpeciation*. *Local optimization* polishes the final models. Flow: In the global optimization phase, every (parallel) island/optimizer globally searches for DNN hyperparameters using a common genotype. Each internal model is trained using a distributed worker (hardware resources) and evaluated for fitness. Periodically, migrations send/receive candidate models to/from the other adjacent islands. Out of all the neighboring models received, one is chosen probabilistically, using rank selection. The selected candidates are transformed into the internal representation of each island/optimizer via transpeciation and replace the worst internal model of the island. Finally, after a number of co-evolutionary iterations, local optimization is performed. Starting point is the best fit model that was co-evolved by global optimization. Local search algorithms use the same search space as the global ones and polish the final DNN model.

IV. METHODS USED FOR OZONE FORECASTING

A. Model

MIMO NARX DNN: Single-station forecasting of air quality has been tried in the past with LSTM [30]. For multi-station predictions, we have the MIMO NARX DNN (Fig. 5):

$$\mathbf{y}(t) = \mathbf{F}(\mathbf{y}(t-1), \mathbf{u}(t)) \tag{7}$$

where t is the time-step (in days), n is the number of O_3 monitoring stations (46 for Belgium), m is the number of exogenous variables (51 weather/environmental variables and 8 calendar cyclical features), $\mathbf{y}(t) \in \mathbb{R}^n$ is the output vector at time t (O_3 concentration at each monitoring station), $\mathbf{y}(t-1) \in \mathbb{R}^n$ the output vector at time $t-1$, $\mathbf{u} \in \mathbb{R}^m$ the exogenous variables vector at time t and $\mathbf{F}(.)$ is the nonlinear vector function (3-layered DNN in our case).

We use only one-day lags because we did not notice any accuracy improvements experimentally by increasing them

Table I
ISLAND SPECIES (OPTIMIZERS) THAT COOPERATE IN ISLAND TRANSPECIATION NAS FOR GLOBAL AND LOCAL SEARCH.

Island Species		
Optimizer	**Internal representation** (of hyperparameters)	**Description**
Global Optimizers		
Random Search (RS)	$[0, 1]$	Uniform distribution sampling, with island population of one [16].
Particle Swarm Optimization (PSO)	position and velocity	Multi-dimensional representation of the parameters as real valued positions, along with rate of change of the positions [11].
Bayesian Optimization (BO)	parameter distribution	Candidates are sampled from a bounded posterior distribution of parameters [15].
Genetic Algorithms (GA)	energy $\in [0, 1]$	Variants used: simple, μ, $\mu+\lambda$, Covariance Matrix Adaptation (CMA) [12], [34].
Differential Evolution (DE)	energy $\in [0, 1]$	Similar to GA [14].
Local (bounded) optimizers		
L-BFGS-B	-	First derivative Quasi-Newton method [35].
SLSQP	-	Sequential Least Squares Programming [36].
TNC	-	Truncated Newton algorithm [37].
Trust-constr	-	Trust-region for unconstrained optimization solving quadratic subproblems [38].

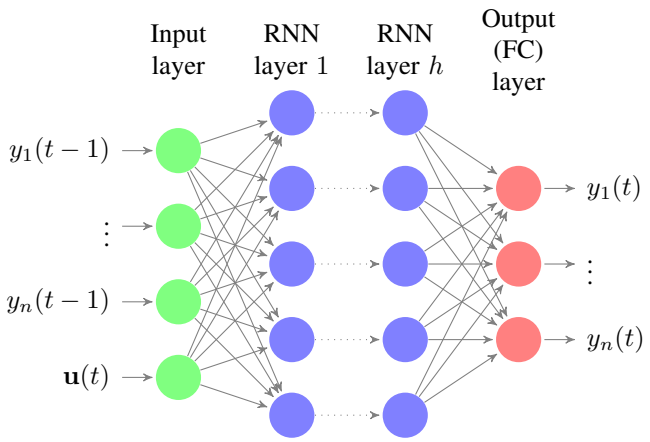


Figure 5. Multiple-Input Multiple-Output (MIMO) Nonlinear AutoRegressive eXogenous (NARX) model, for next-day air-quality forecasting: $\mathbf{y}(t) = \mathbf{F}(\mathbf{y}(t-1), \mathbf{u}(t))$. Predict next-day (time t) air-quality for n measuring station outputs $\mathbf{y}$, using exogenous variables $\mathbf{u}(t)$ (e.g., weather forecasts) and previous-day measurements $\mathbf{y}(t-1)$. Recurrent Neural Networks (RNN) act as h hidden layers and the final Fully Connected (FC) layer provides the output numerical predictions.

further. In addition, we had an increment (almost double or more per extra delay) in dataset size and model training times. We used daily time-step instead of hourly for two reasons: (1) our exogenous weather data had a minimum time-step of 3 hours and (2) because the environmental agencies like the Belgian Interregional Environment Agency (IRCEL - CELINE) usually inform the public with 24-hour means. Fig. 6 illustrates an example *MIMO NARX DNN* architecture. The

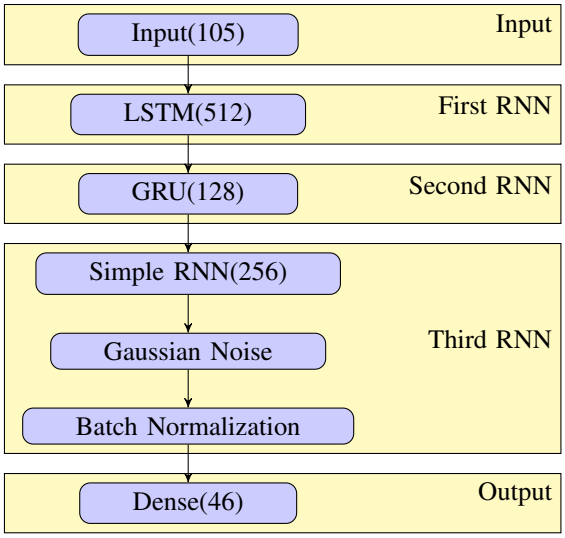


Figure 6. Illustration of an example *MIMO NARX DNN* architecture, evolved by island transpeciation for next-day O_3 forecasting of 46 measuring stations. 105 inputs (51 exogenous variables, 8 calendar features and previous-day air-quality measurements from 46 stations), three Recurrent Layers (512 neuron LSTM, 128 neuron GRU, 256 neuron Simple RNN), two auxiliary layers (Gaussian Noise, Batch Normalization) and one Dense Layer with 46 outputs (next-day air-quality predictions for 46 O_3 stations). Note: layer types and hidden neuron sizes shown are arbitrary, a depiction of the architecture evolution possibilities.

layer types and hidden neuron sizes shown are arbitrary, a depiction of the architecture evolution possibilities.

B. Search space

Representation: For the architecture and hyperparameter search space representation, upper and lower bound arrays were used as constraints. The arrays are of fixed length and contain integer (nominal) or float value ranges. The genotype of a model is a value set, sampled from this bounded space. The candidate model phenotypes are the trained sequential DNN models, using the architecture defined by the genotype. The models were generated and compiled using the Tensorflow framework [39].

The candidate models are fixed to three base recurrent layers (see Table II). Architecture search genes are expressed as bounded integer values. They represent: (1) the type of the core recurrent layer (LSTM [8], simple RNN [6] or GRU [9]) and (2) the recurrent layer sequence causality (simple or bidirectional [7]). The auxiliary/utility layers are placed between the base layers. Auxiliary placements are determined by the evolutionary search. Auxiliary layer types can be: (1) Batch Normalization [40] and (2) Gaussian Noise [41]. Layer weight initializer genes (Table II), determine the sampling distribution of the initial neuron weights. *Optimizers* are the algorithms that guide the weight training of an Artificial Neural Network (ANN) [42]. We used all the available optimizers that work with recurrent DNN architectures [43]. Note that in this paper an ample amount of hyperparameters are considered in the search space. The reasoning behind this is twofold: First, to demonstrate the proposed solution in a large search space setting. Second, restricting the hyperparameter search

Table II
DISCRETE HYPERPARAMETER BOUNDS (ARCHITECTURE SEARCH GENES IN THE COMMON GENOTYPE).

Hyperparameter	Values
Layer	
Simple or Bidirectional	LSTM, RNN, GRU.
Auxiliary	Gaussian Noise, Batch Normalization.
Weight Initializer	Zeros, uniform (random, lecun, he), ones, normal (random, lecun, he, glorot, truncated).
L1/L2 weight decay	Activity regularizers, bias regularizers, kernel regularizers.
Deep Neural Network	
Optimizer	Adam [44], nadam [45], amsgrad [46], adagrad [47], adadelta [48].

Table III
NUMERICAL HYPERPARAMETER BOUNDS (FOR EACH DNN LAYER).

Hyperparameter	Min	Max	Description
batch size	7	31	Samples per model update.
epoch size	350	600	Training data pass count.
units	64	512	Neuron count per layer.
dropout	0.01	0.25	Unit fraction to leave-out.
recurrent dropout	0.01	0.25	Recurrent unit fraction to leave-out.
gaussian noise STD	0.1	0.5	Noise distribution standard deviation.
L1, L2 regularizers	0.0	0.01	L1 and L2 layer optimization penalty.

too much could result in a sub-optimal model. In practice one could reduce the search space by expert knowledge about the problem at hand.

The continuous floating-point value bounded parameters that we optimized, are shown in Table III. We chose the hyperparameter bounds both empirically and from suggested ranges on previous works [49], [50], [51]. Mini-batches increase DNN model generalization [51] but we have to accept that model training times will increase. We chose a minimum batch size of 7 (days), because a week time-span may exhibit cyclical calendar patterns. The epoch size parameter acts mostly as a max range in our setup. We empirically chose large ranges, since we also use early (model training) stopping [43] to avoid overfitting. Due to model memory constraints, we chose to have at maximum three RNN layers with at most 512 units each. In terms of dropout and recurrent dropout, suggested values range from 0.1 up to 0.5 [49], [50]. However, the upper ranges are mostly destined for very large models, so we empirically chose a conservative range of max 0.25. Gaussian noise [52] and regularizers [50] improve model generalization. Again, we chose noise bounds empirically.

C. Search strategy

In the ring topology, each island is a subpopulation that connects virtually and directly to another subpopulation using one dimension. Grid island or cellular topologies [32] of two dimensions can be folded into a torus. Due to high graph connectivity, a torus interconnect architecture [53] can have vast amounts of optimization islands, while requiring very few hops to propagate messages through the whole structure (Fig. 7). Solution diversity is maintained for multiple generations and the most performant models will not rapidly dominate the island structure. The neighborhoods can be expanded to four dimensions (hypercubes) or more.

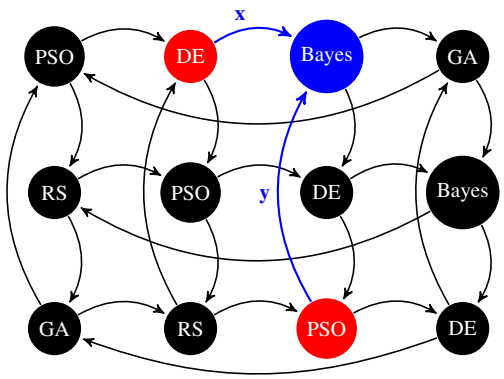


Figure 7. Adjacent island structure in 2D island transpeciation: Single-direction Von Neumann neighborhood (radius 1) of a two-dimensional torus interconnect architecture. The *Bayes* optimization island receives adjacent demes (candidate solutions) from the *DE* (x dimension) and the *PSO* (y dimension) islands. By constraining the migration direction, we achieve a gradual, "wind-like" wave of propagation of solutions, instead of an uncontrolled "wild-fire" spread.

The islands iterate asynchronously. Training and evaluation times depend on the count of the trainable parameters of the model. In addition, every optimization algorithm-island does not iterate at the same pace. Removing any implicit or explicit synchronization barriers after model training and evaluation, allows the evolved models to complete not only in accuracy but also training speed. The removal of synchronization barriers allows for the usage of the competing consumers pattern, which is discussed in Appendix II-A. Migrations of candidate solutions can still occur but after a predefined count of island iterations (migration interval). In our case, each island receives migrating candidate solutions (demes) from its neighbors every 5 fitness evaluations.

An n-dimensional grid yields μ neighbors per island. But which migrating candidate to accept? We used *Linear Ranking (LR) selection* [54]:

1) Rank μ neighbors by fitness (worst rank: i = 0).
2) Adjust selection pressure $s \in (1, 2]$:
   - Small $s \Rightarrow \approx$ uniform random.
   - Large $s \Rightarrow$ best ranks have higher selection probability.
3) Random choice given probabilities:

$$P_{LinearRank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)} \qquad (8)$$

Finally, the island's worst candidate is *replaced* with the newly selected, at the "internal representation" level. This is a fitness-based replacement strategy known as *replace worst* (GENITOR) [55]. The negative effect of the replace worst strategy, is that there may be premature convergence and genotype duplicates inside a population. The positive effect is that the mean fitness of the whole population is rapidly increased.

Phenotypic plasticity is the adaptation of an individual to a specific environment [56]. These adaptations are temporary, act only during a lifetime and do not propagate to offspring. To achieve a similar effect on DNN NAS, we applied random *phenotypic mutations* [57]. Naturally, seemingly similar DNNs can have vastly different performance due to neuron weight training randomness. With phenotypic mutations we can introduce additional DNN model variations, not guided by evolution or the common genotype. The bounded architecture search space is expanded from 25 variables to 53: we included layer-based regularization hyperparameters. These regularizers apply additional penalties to the loss function. L1 and L2 norm weight decay penalty is randomly applied (sampled from a uniform distribution) during weight optimization (Table II), with: activity, bias and kernel regularizers.

D. Performance estimation strategy

To compare forecasting model performance in NAS, we used two metrics. First, the symmetric Mean Absolute Percentage Error (sMAPE) [58], formally as:

$$sMAPE = \frac{2}{T}\sum_{t}^{T}\frac{|y(t)-\hat{y}(t)|}{|y(t)|+|\hat{y}(t)|} * 100(\%) \quad (9)$$

where t is timestep of maximum T, $\hat{y}(t)$ the forecast and $y(t)$ the observation at time t. Second, the Mean Absolute Scaled Error (MASE) [58]:

$$MASE = \frac{1}{J}\sum_{j}^{J}|\frac{y(j)-\hat{y}(j)}{\frac{1}{T-m}\sum_{t=m+1}^{T}|y(t)-y(t-m)|}| \quad (10)$$

where j is forecast period out of total forecasts J and m is the seasonal period (with $m = 1$ for non-seasonal Naive-1 forecast error).

V. Results

A. Description of the experiment

Real-world, publicly available data are used in all experiments. Ozone (O_3 max daily 8-hour rolling mean concentrations) data were gathered from the European Environmental Agency (EEA) [3]. Data was augmented with 51 weather variables (see Table IV in Appendix B), acquired from the surface-level ERA-interim ECMWF public database [4], [26]. The first set of air quality data from EEA, start from 1990 and end up in 2011. The dataset contains daily, hourly and yearly values. For the years 2012 to 2018 the data provided are only hourly, which require pre-processing in order to be converted to daily. In addition, all the time-series were standardized on the training phase and unstandardized for the predictions. Extra information of the implementation is given in Appendix A.

For the initial MISO models, we used as input 51 exogenous weather variables (see Table IV in Appendix B) and a single monitoring station time-series with 1-lag (1 day before). The output is the forecast for the current day. The MIMO models use the same inputs, but the output is the current day forecast of all the time-series. All the data available for Belgium are used. For the MIMO and MISO models predicting 2018, we added 8 calendar cyclical features (*sine* and *cosine* of: month, day of week/year, week of year). Fig. 8 shows the geo-locations of all the 46 O_3 stations used in the experiments. This view contains all the station types: urban, suburban, rural and rural background.

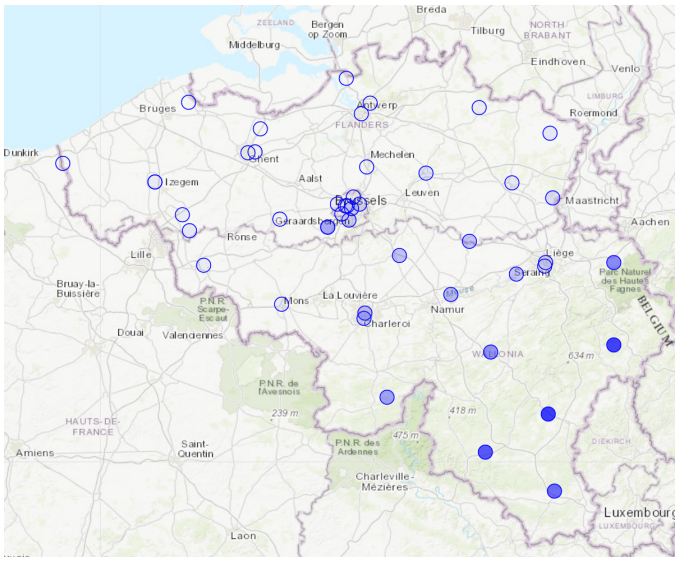


Figure 8. All the available 46 O_3 monitoring station locations that we used for country-scale (Belgium) air-quality forecasting. The full time-series dataset (1990 to 2018) is publicly accessible from the European Environmental Agency (EEA) [3].

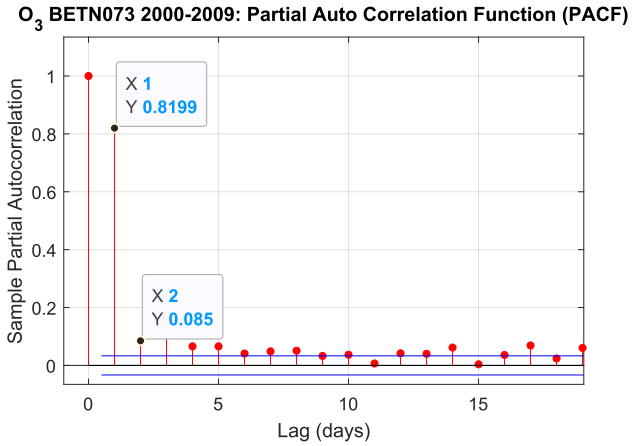


Figure 9. Partial AutoCorrelation Function (PACF) for the O_3 background-rural station measurements (BETN073 at Andenne, Belgium: 2000 to 2009). There is high partial autocorrelation for 1-day lag (0.82) but less significant (0.09) for 2-day lags or more.

The datasets have two significant limitations. First, the exogenous variables are only sampled by 1 location in the middle of Belgium. The reason is data scaling issues. Adding up to 46 stations x 51 variables would significantly increase the model training times of the experiments. In future work we will use local weather information, along with feature reduction from hyperparameter tuning and expert knowledge. Second, only 1-day lag was used in all cases. We tested adding one extra lag-day, but we did not notice any increment in

accuracy, while we almost doubled the amount of training data. The Partial AutoCorrelation Function (PACF) for O_3 (Fig. 9) shows high partial autocorrelation for 1-day lag (0.82) but much lower (0.09) for 2-day lags or more.

For pre-processing, ECMWF performs 4D-variational data assimilation on the weather data. EEA flags air-quality measurements as "verified" if they pass full quality assurance and quality control and "valid" if values are above detection limit and not due to station maintenance or calibration. We removed any invalid/unverified data and filled them with values from the most correlated stations time-series of the same timestep. For any remaining missing values, we applied linear interpolation (forward/backward fill). Hourly values, were aggregated as daily max of 8-hour rolling mean. Finally, features are standardized.

To avoid model overfitting and allow for better generalization for time-series predictions, we applied a variant of cross-validation: Time-series Cross-Validation [59]. This technique does not shuffle the data sequence and also there is no information "spill-over" from future steps. The mean MSE of a 5-fold cross-validation determines whether a model is optimal and generalizes well. The "test" data are held-out from the start, never used in model training. Next to that, we employ *Early Stopping* and *Learning-Rate reduction on plateau* [43]. In case there are no improvements in the global island model fitness search, after a specific amount of iterations, the master process can abort the searching operation and stops all the other island processes. This can act as an additional stopping criterion for the evolutionary search, along with the max allowed iterations setting (set to 500). In our experiments, we disabled the automatic stopping in order to examine the co-evolution for around 350 to 500 iterations.

B. Results for MISO models

1) Island transpeciation versus random search: With the GIM model [23], it was shown that combinations of different global hyperparameter optimizers in island structures, can outperform single optimizers. For the context of air-quality, we used a subset of the real-world O_3 measurements and halved the max neuron count per layer, to compare island transpeciation in NAS against a commonly found optimizer, random search.

Fig. 10 shows the evolution of model accuracy during NAS. The horizontal axis denotes the count of iterative model improvements and the vertical axis the test accuracy. Each scenario was repeated five times. Out of 2500 models trained per case, island transpeciation found a better model (76.943%) than RS (76.896%). Also, island transpeciation approaches global maxima $\approx$ 150 iterations earlier than RS. For the first $\approx$ 200 iterations, RS had consistently worse performance than island transpeciation. We noticed the same pattern with a similar experiment repeated on 2012 as the test year and 1/4 the max allowed DNN model size. Island transpeciation setup was: 18x optimization islands (4 x GA, 4 x BO, 4 x DE, 3 x RS, 3 x PSO) on a 3x3x3 Cellular Automata grid, 3 directional neighborhood, 500 iterations (350 global search + 150 for local search polishing).

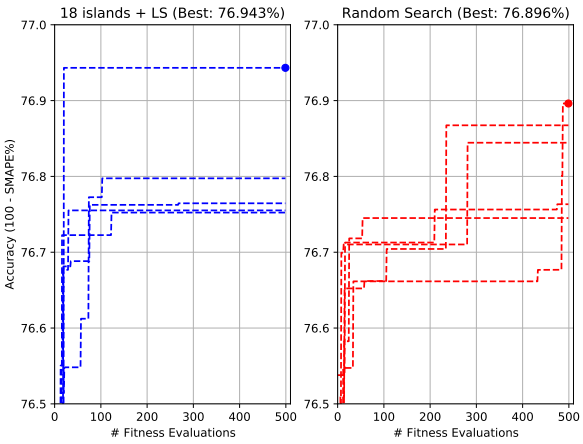


Figure 10. Island transpeciation (left) versus random search (right) NAS, on a real-world O_3 data subset: Island transpeciation generated more accurate models (best: 76.943%, mean: 76.75% +/- 0.55%, model samples: 2500) and converges ($\approx$ 150 iterations) earlier than Random Search (best: 76.896%, mean: 76.72% +/- 0.26%, model samples: 2500). Each NAS search was repeated 5 times, for 500 iterations each. Target model type is MISO NARX, for the background-rural station BETN073 (code-name for the Belgian city Andenne). Train: 2000 to 2009, test: 2010, exogenous data: 6x weather variables, three cross-validation folds.

2) Island DNN versus other models: Fig. 11 compares one-day-ahead O_3 forecasting MISO models. "Naive-1" at the bottom is the baseline and simplest model: the forecast is the value of the previous day. So, the baseline Mean Absolute Scaled Error (MASE) is 1. We tried additional models but we show only the ones that surpassed the baseline. The Island DNN at the top was optimized by island transpeciation, both in architecture and hyperparameters. It was the best model and achieved the lowest MASE (0.655). Most of the other model types were trained with Matlab Regression learner, 10-fold cross-validation, regularization and 350 iteration hyperparameter Bayesian optimization. The Support Vector Machines (SVM) had a medium gaussian kernel and performed the worst with a MASE of 0.802. The best Gaussian Process Regression (GPR) had an rational quadratic kernel and performed better (MASE: 0.737) than the plain SVM. Least-Squares Support Vector Machines (LSSVM) [60] are reformulations to the standard SVMs and exploit primal-dual interpretations. LSSVM was the third best model (MASE: 0.705) and the second best was the Tree Ensemble model (MASE: 0.678). One argument against DNN is that they require long training times. However, if one accounts for the hyper-parameter optimization process, we noticed that the other model types did require multi-hour/day training also (especially GPR). In addition, MISO NARX DNN can be easily expanded to MIMO, in order to forecast multiple measuring stations concurrently. To our knowledge, this is not readily possible for most of the other model types. Adding one or several extra outputs would also considerably increase the convergence training times for the other model types, but not for the DNN case.

Fig. 12 shows the prediction output of the best model, the

optimized MISO NARX DNN model. The horizontal axis denotes time, 365 days of the year 2010, with a day as a time-step. The vertical axis denotes the sensor value for an O_3 station: the background-rural station BETN073 (code-name for the Belgian city Andenne). The blue line is the ground truth, the sensor values at the monitoring station. The orange line shows the next-day O_3 prediction. Island transpeciation setup was: 18x optimization islands (4 x GA, 4 x BO, 4 x DE, 3 x RS, 3 x PSO) on a 3x3x3 Cellular Automata grid, 3 directional neighborhood, 500 iterations (350 global search + 150 for local search polishing).

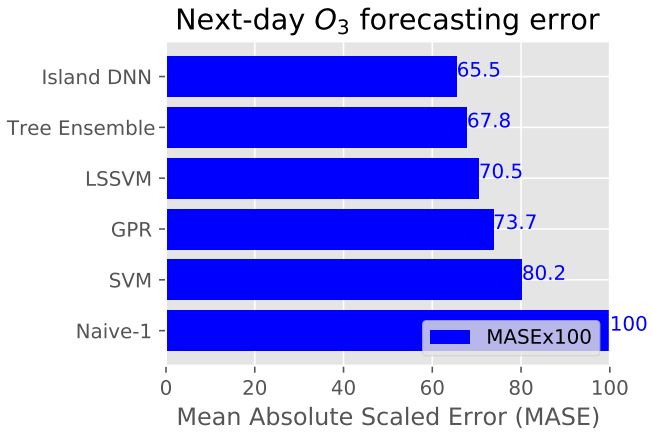


Figure 11. Next-day O_3 background-rural station (BETN073 at Andenne, Belgium) forecasting error: Baseline (previous day in-sample value: Naive-1) against the best DNN, GPR, SVM and LSSVM MISO NARX models. Most models were trained with Matlab Regression learner, 10-fold cross-validation, regularization and 350 iteration hyperparameter Bayesian optimization. Island transpeciation NAS trained the Island DNN, which had the lowest error. Island transpeciation setup: 18x optimization islands (4 x GA, 4 x BO, 4 x DE, 3 x RS, 3 x PSO) on a 3x3x3 Cellular Automata grid, 3 directional neighbors, 500 iterations (350 global search + 150 for local search polishing). Training: 2000 to 2009, test: 2010, with 51 weather exogenous variables and linear interpolation on missing values.

C. Results for MIMO models

Fig. 13 shows observations and predictions of 25 O_3 monitoring stations for the year 2012, projected one on top of the other. It seems counter-intuitive to show a collective view; the reason is to show the spike around day 210. During the summer of 2012, there was an "inform public" alert, which happens when more than one stations have a measurement greater than 180 $\mu g/m^3$. The MIMO island DNN predicted the alert level successfully.

D. MISO versus MIMO models

Fig. 14 compares the MISO versus MIMO models for single-station prediction. We want to predict one station, BETN073 (background-rural O_3 station). We compare three models: (1) "1-station model", (2) "all-stations model" (inputs from all the 46 O_3 stations in Belgium) and (3) "background-rural-stations model" (inputs from only the 18 background-rural O_3 stations). The "background-rural-stations model" had the worst performance in all metrics. The "1-station model" was mediocre initially, but improved vastly after 500 island transpeciation iterations. The additional 8 cyclical calendar features (sine and cosine of day of the week/month/year) helped in accuracy even more. The "all-stations model" was the most accurate: reduced the symmetric Mean Absolute Percentage (sMAPE) error by 1.45% against the single-station and 6.26% against the background-rural stations model.

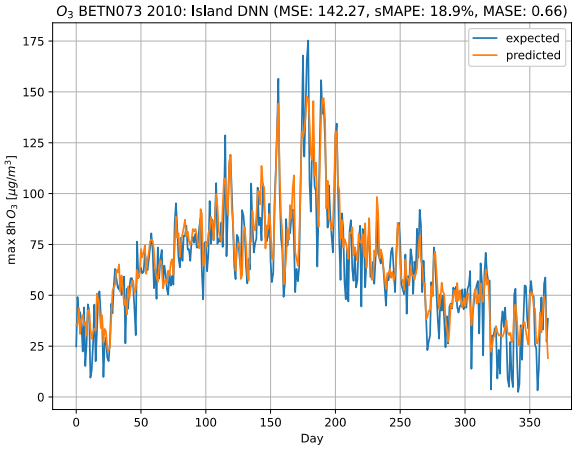


Figure 12. One day ahead predictions of O_3 (single-station): Multiple-Input Single-Output (MISO) NARX DNN, optimized by island transpeciation NAS. The horizontal axis denotes time, 365 days of the year 2010, with a day as a time-step. The vertical axis denotes the sensor value for an O_3 station, specifically the background-rural station BETN073 (code-name for the Belgian city Andenne). The blue line is the ground truth, the sensor values at the monitoring station. The orange line shows the model prediction. This model outperforms the other ML approaches. Train: 2000 to 2009, test: 2010, data: 51x weather variables [4].

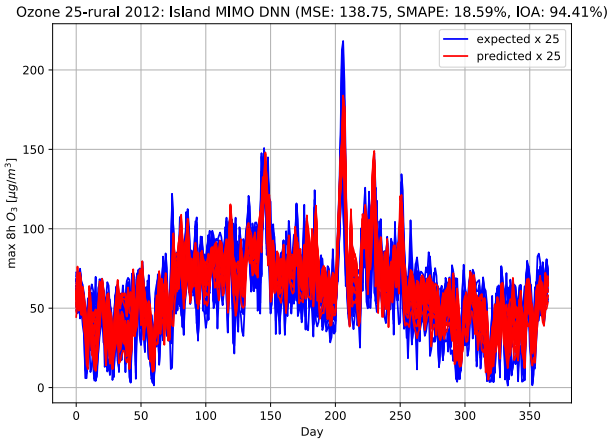


Figure 13. One day ahead predictions of O_3 (25 Belgian stations) with a MIMO NARX DNN optimized by island transpeciation: At day ≈210 our model successfully predicted a real-world incident, the **"inform public" level** one-day earlier. Train: 1990 to 2011, test: 2012.

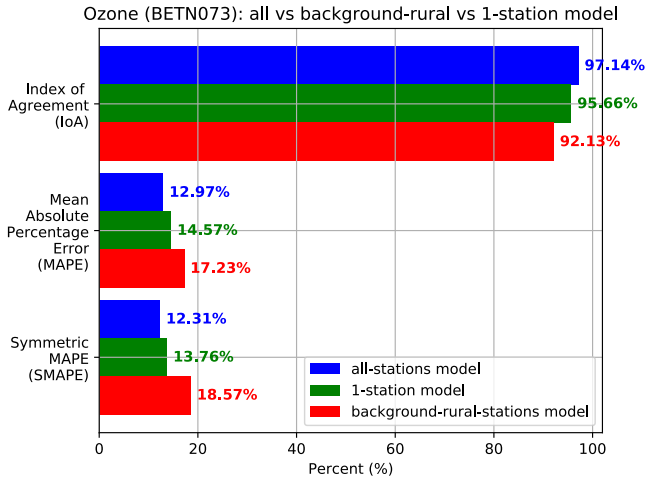


Figure 14. Single O_3 station (BETN073) prediction comparison: all-station vs 1-station model vs background-rural. All the models are NARX DNN, optimized by island transpeciation NAS. "1-station model": MISO NARX trained only with the BETN073 station time-series and weather/calendar data. "background-rural-stations model": MIMO NARX trained only with the O_3 background-rural stations (18 total) of Belgium. "all-stations model": MIMO NARX trained with all the stations O_3 of Belgium (46 total). The all-stations model was the most accurate. Train: 2010 to 2017, test: 2018.

E. Island transpeciation as NAS

We will see a case of global optimization with island transpeciation for O_3. In Fig. 15 we illustrate the NAS model accuracy progression when trying to find a suitable architecture for a MIMO NARX DNN model. The vertical axis denotes the model accuracy on the test set and the horizontal the total number of fitness evaluations. Each dot type represents DNN accuracy from different island/hyperparameter optimizers. The blue line at the top shows the best found model from the whole archipelago. The dashed line shows the accuracy trend of all the models.

We used 18 island variants of the following global optimizers: Bayesian optimization, random uniform search, differential evolution, particle swarm optimization and genetic algorithms. The global optimizers were organized into a 1-dimensional ring of 18 slots. Every island had one neighbor. NAS overall had a positive trend, during the ≈500 total fitness evaluations. The first ≈400 iterations were devoted to global search and the last 100 for local search. The local search was done by three non-communicating islands, running different local optimizers. Overall, the median model had 90.74% +/- 2.6% accuracy ($100 - sMAPE\%$). There was a vast range in model performance (worst 74.01% and best 92.60%), which further shows the need to optimize DNNs. If we would have used only GA optimizers (blue triangles), DNN model improvements would stop at 200 fitness evaluations: GA model performance starts to degrade after ≈ 150. However, with island transpeciation and the migration of models between different optimizers, GA is able to provide improvements again after 300 fitness evaluations. Local Search polishes the best globally optimized models for the final ≈ 100 iterations. We can see that island transpeciation NAS indeed generated a consistently performant architecture because the final fitness evaluations had similarly high accuracy.

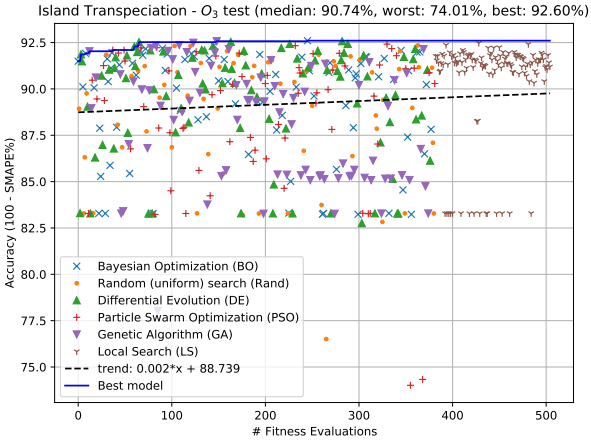
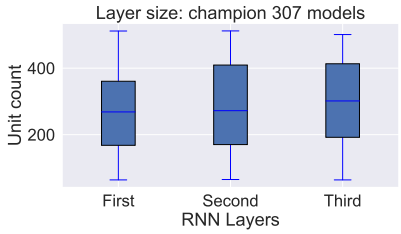


Figure 15. Accuracy progression with island transpeciation NAS on O_3 2018 (all 46 stations). The global optimizer islands were organized into a 1-dimensional ring of 18 slots. Every island had one neighbor. For every iteration the DNN was trained from scratch, using the evolved architecture. The first ≈400 iterations were devoted to global search and the last 100 for local search. The models had a vast accuracy range: worst 74.01% and best 92.60%. 18 islands (1D grid: 1x18), 5 individuals per island, calendar cyclical features. Local Search (LS) iterations had similarly good accuracy (close to the best), showing that at the end island transpeciation indeed generated performant DNN architectures.

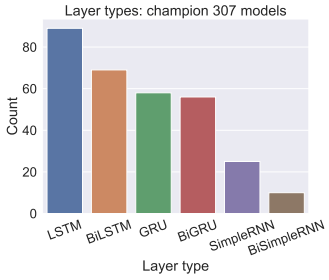
F. Champion NAS models

To examine the MIMO NARX DNN models in terms of architecture and hyperparameters, we will aggregate all the 307 champion genotypes that were evolved from multiple experiment NAS runs (total 6659 models). Fig. 16 shows results from three important hyperparameters: layer size, layer type and DNN optimizer.

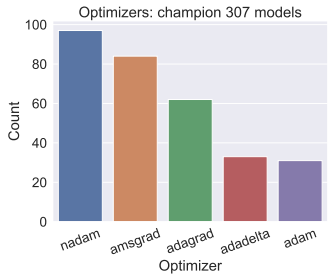
The champion models (Fig. 16a) had a layer unit count close to the middle of the bounded space (64 to 512 units).With island transpeciation, models compete both on accuracy and model training speed, which auto-regulates the "survival of the fattest" (the tendency in GA where average model size grows continuously) [61]. Models with the maximum trainable parameters have higher learning capacity, but they can be slow to train. Smaller models may perform worse than the large ones initially, but they can be trained faster and evolve more. Experiencing more iterative improvements may yield better architectures. In Fig. 16b we examine the recurrent layer types selected by the optimizers. The most selected was the LSTM, almost 4 times more than the bidirectional simple RNN layers. GRU layers stand in the middle of the selection preference. The bidirectional versions were chosen the least because their capabilities were not fully utilized: no future time-series steps were provided. Even though the core recurrent layers return full sequences between them, there was no tangible benefit. In addition, all the models had only inputs of 1 slice, lag of 1 day. Fig. 16c shows that most of the champion models had



(a) Units hyperparameter (layer size in neurons).



(b) Layer type hyperparameter.



(c) DNN optimizer hyperparameter.

Figure 16. Top 307 (out of 6659) island transpeciation champion model hyperparameters. Layer size (a): The champion models had a unit count close to the middle of the bounded space (64 to 512 units). Larger models with more trainable parameters have higher learning capacity, but they can be slow to train. Smaller models may perform worse but, they can be trained faster and experience more iterative improvements by the global co-evolution. With the implicit evolutionary pressure of the "model size vs training speed" trade-off, we can find performant and accurate DNN architectures, instead of architectures with the max trainable parameters possible ("survival of the fattest" [61]). Layer types (b): LSTM was selected about 4 times more than the bidirectional simple RNN layers. GRU layers stand in the middle of the selection preference. The plain versions were selected more often than the Bidirectional (Bi prefix) variants. DNN (weight training) optimizers (c): Most of the champion models had nadam and amsgrad. Adam and adadelta were preferred the least.

nadam and amsgrad as DNN weight optimizers. Adam and adadelta were not preferred as often.

VI. Conclusion

Multiple-Input Multiple-Output (MIMO) Nonlinear Auto-Regressive eXogenous (NARX) Deep Neural Networks (DNN) can predict next-day O_3 pollution episodes, at a country-scale. This "all-in-one" modelling approach surpassed single-model performance on real-world, publicly available data: We used time-series (1990 to 2018) from 46 O_3 Belgian monitoring stations (European Environmental Agency) that we combined with 51 weather variables acquired from the surface-level ERA-interim European Centre for Medium-Range Weather Forecasts (ECMWF) database. MIMO NARX DNN successfully predicted one day before, an "inform-public" O_3 alert level in Belgium for 2012. Predictions improved with: data standardization, time-series split (cross-validation variant), adding weather/atmospheric variables and cyclical calendar features. The main negative is that they require long training times, especially if we increase the model's lag count.

"Island transpeciation", optimized MIMO NARX DNN hyperparameters and architectures that outperformed random model search and other previous modelling efforts. Unlike single "off-the-shelf" optimizers, island transpeciation can combine results from multiple optimizers, to consistently provide excellent performance. Island transpeciation is a co-evolutionary meta-learning method that combines Neural Architecture Search, neuroevolution and global/local optimizers. With cooperation and competition, the "survival of the fattest" side-effect of meta-learning (*model size* versus *training speed* trade-off) is auto-regulated, via the asynchronous Cellular Automata distributed communication. Heterogeneous hardware resources can be parallelized with fault tolerance and distributed control. However, island transpeciation requires additional configuration, compared to "off-the-shelf" hyperparameter optimizers. The encoding/decoding of internal representations to genotypes can be lossy, requires altering the optimizer source code internals and it is not always bidirectional (e.g., random search can not algorithmically accept model migrations). Finally, adding islands increases the computational parallelism needs. Islands should iterate enough in order for the underlying optimizers to perform sufficiently.

Appendix A
Development and Repositories

A. CPU and Multi-GPU parallelism

Initially, global optimization species are spawned in parallel in the CPU. Distributed workers connect to the islands via a message broker. The island architecture uses a (blocking) master-slave [62] pattern (Fig. 17): Island migration is carried out by slaves (islands) sending candidates to the master (main CPU process). The master process keeps a buffer of all the best candidates of each island. Every 5 generations (the migration interval μ), an island sends its best candidate model to the master process, while receiving a champion candidate (randomly picked using linear ranking selection) from the neighboring islands. Then, the island replaces its worst internal model with the candidate that just received from the neighborhood. Message Passing Interface (MPI) [63] handles the communication between the islands and the master.

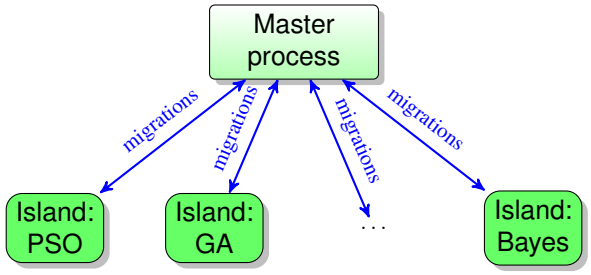


Figure 17. CPU parallel implementation for island-to-island model migrations: Master-slave pattern. Message Passing Interface (MPI) [63] handles the communication.

For DNN model training, the asynchronous "Competing Consumers" pattern was utilized [64], using the Rabbit MQ library [65] (Fig. 18) as a message broker. Networked heterogeneous workers (GPU hardware of different training performance) and architecture were kept "always busy": NVidia GTX 970 4GB with Maxwell architecture and NVidia GTX 1070 Ti 8GB with Pascal architecture) were able to cooperate

seamlessly. The parallel implementation was tested simultaneously: on a local workstation (6 physical CPU cores, 2x Cuda capable GPUs), on the Amazon Elastic Compute Cloud (EC2) with p3.8xlarge Amazon Machine Image (AMI) containing 4x Tesla V100 GPUs and on nodes of 4 NVidia Tesla P100 GPUs from the Vlaams Supercomputing Centrum (VSC) [66].

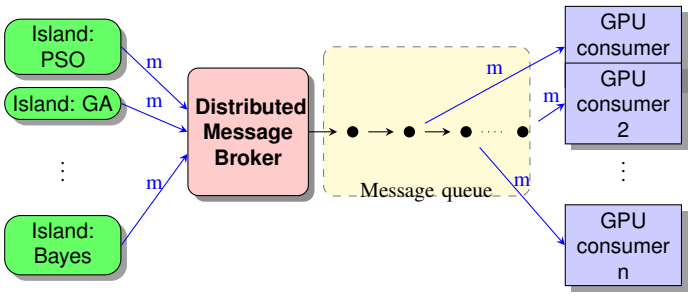


Figure 18. Multi-GPU distributed implementation for DNN model training: Asynchronous competing-consumers pattern (with m denoting the model genotype to train). The message broker [65] keeps a queue of island DNN models to be trained in sequence. Networked GPU heterogeneous consumers, train models without synchronization delays and explicit barriers. The queue is fault-tolerant by using acknowledgments and heart-beat timeout messages. At any time during the island transpeciation training process, it is possible to add or remove GPU hardware consumer nodes (hot-plugging).

B. Repositories

- Project: https://github.com/temp3rr0r/Ozone-Narx-DNN
- Weather, atmospheric and measuring station data: https://github.com/temp3rr0r/Ozone-Narx-DNN/tree/master/models/NarxModelSearch/data
- Experiment runs: https://github.com/temp3rr0r/Ozone-Narx-DNN/tree/master/models/NarxModelSearch/runs

APPENDIX B
EXOGENOUS VARIABLES

See Table IV.

Table IV
EXOGENOUS WEATHER VARIABLES [4], [26] (SURFACE-LEVEL ERA-INTERIM ECMWF PUBLICLY-AVAILABLE DATABASE)

ID	Short name	Name	Unit
20	parcs	Clear sky surface photosynthetically active radiation	$J \cdot m^{-2}$
31	ci	Sea ice area fraction	$(0-1)$
33	rsn	Snow density	$kg \cdot m^{-3}$
34	sst	Sea surface temperature	K
35	istl1	Ice temperature layer 1	K
36	istl2	Ice temperature layer 2	K
37	istl3	Ice temperature layer 3	K
38	istl4	Ice temperature layer 4	K
39	swvl1	Volumetric soil water layer 1	$m^3 \cdot m^{-3}$
40	swvl2	Volumetric soil water layer 2	$m^3 \cdot m^{-3}$
41	swvl3	Volumetric soil water layer 3	$m^3 \cdot m^{-3}$
42	swvl4	Volumetric soil water layer 4	$m^3 \cdot m^{-3}$
49	10fg	10 metre wind gust since previous post-processing	$m \cdot s^{-1}$
50	lspf	Large-scale precipitation fraction	s
59	cape	Convective available potential energy	$J \cdot kg^{-1}$
134	sp	Surface pressure	Pa
136	w	Total column water	$kg \cdot m^{-2}$
139	stl1	Soil temperature level 1	K
141	sd	Snow depth	m of water equivalent
142	lsp	Large-scale precipitation	m
143	cp	Convective precipitation	m
144	sf	Snowfall	m of water equivalent
146	sshf	Surface sensible heat flux	$J \cdot m^{-2}$
151	msl	Mean sea level pressure	Pa
159	blh	Boundary layer height	m
164	tcc	Total cloud cover	$(0-1)$
165	10u	10 metre U wind component	$m \cdot s^{-1}$
166	v	10 metre V wind component	$m \cdot s^{-1}$
167	2t	2 metre temperature	K
168	2d	2 metre dewpoint temperature	K
169	ssrd	Surface solar radiation downwards	$J \cdot m^{-2}$
170	stl2	Soil temperature level 2	K
175	strd	Surface thermal radiation downwards	$J \cdot m^{-2}$
176	ssr	Surface net solar radiation	$J \cdot m^{-2}$
179	ttr	Top net thermal radiation	$J \cdot m^{-2}$
182	e	Evaporation	m of water equivalent
183	stl3	Soil temperature level 3	K
186	lcc	Low cloud cover	$(0-1)$
187	mcc	Medium cloud cover	$(0-1)$
188	hcc	High cloud cover	$(0-1)$
198	src	Skin reservoir content	m of water equivalent
201	mx2t	Maximum temperature at 2 metres since previous post-processing	K
202	mn2t	Minimum temperature at 2 metres since previous post-processing	K
205	ro	Runoff	m
206	tco3	Total column ozone	$kg \cdot m^{-2}$
228	tp	Total precipitation	m
229	iews	Instantaneous eastward turbulent surface stress	$N \cdot m^{-2}$
235	skt	Skin temperature	K
236	stl4	Soil temperature level 4	K
243	fal	Forecast albedo	$(0-1)$
244	fsr	Forecast surface roughness	m

ACKNOWLEDGMENTS

EU: The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. Research Council KUL: Optimization frameworks for deep kernel machines C14/18/068. Flemish Government: FWO: projects: GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant. This research received funding from the Flemish Government (AI Research Program). Konstantinos Theodorakos, Oscar Mauricio Agudelo, Joachim Schreurs, Johan Suykens, Bart De Moor are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms).

KU Leuven: Research Fund (projects C16/15/059, C3/19/053, C24/18/022, C3/20/117), Industrial Research Fund (Fellowships 13-0260, IOF/16/004) and several Leuven Research and Development bilateral industrial projects. Flemish Government Agencies, FWO: EOS Project no G0F6718N (SeLMA), SBO project S005319N, Infrastructure project I013218N, TBM Project T001919N; PhD Grants (SB/1SA1319N, SB/1S93918, SB/1S1319N), EWI: the Flanders AI Research Program, VLAIO: Baekeland PhD (HBC.20192204) and Innovation mandate (HBC.2019.2209), CoT project 2018.018. European Commission: European Research Council under the European Union's Horizon 2020

research and innovation programme (ERC Adv. Grant grant agreement No 885682). Other funding: Foundation 'Kom op tegen Kanker', CM (Christelijke Mutualiteit).

REFERENCES

[1] "Health Effects Institute. State of Global Air 2019," 2019. [Online]. Available: www.stateofglobalair.org

[2] C. Guerreiro, F. de Leeuw, A. G. Ortiz, M. Viana, and A. Colette, "Air quality in Europe — 2018 report | EEA," 2018.

[3] "AirBase - The European air quality database." [Online]. Available: https://www.eea.europa.eu/data-and-maps/data/airbase-the-european-air-quality-database-8

[4] D. P. Dee, S. M. Uppala, A. J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. A. Balmaseda, G. Balsamo, P. Bauer, P. Bechtold, A. C. M. Beljaars, L. van de Berg, J. Bidlot, N. Bormann, C. Delsol, R. Dragani, M. Fuentes, A. J. Geer, L. Haimberger, S. B. Healy, H. Hersbach, E. V. Hólm, L. Isaksen, P. Kållberg, M. Köhler, M. Matricardi, A. P. McNally, B. M. Monge-Sanz, J.-J. Morcrette, B.-K. Park, C. Peubey, P. de Rosnay, C. Tavolato, J.-N. Thépaut, and F. Vitart, "The ERA-Interim reanalysis: configuration and performance of the data assimilation system," *Quarterly Journal of the Royal Meteorological Society*, vol. 137, no. 656, pp. 553–597, 4 2011.

[5] K. Theodorakos, *Air-quality forecasting in Belgium using Deep Neural Networks, Neuroevolution and distributed Island Transpeciation*. M.Sc. thesis. Katholieke Universiteit Leuven. Faculty of Engineering Science. Department of Electrical Engineering. ESAT-STADIUS, 9 2019.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1724–1734.

[10] H. Xie, H. Tang, and Y. H. Liao, "Time series prediction based on narx neural networks: An advanced approach," in *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, 2009, pp. 1275–1279.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[12] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagńe, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, no. 70, pp. 2171–2175, 2012.

[13] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. IEEE, 12 2016, pp. 261–265.

[14] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[15] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2012, pp. 2951–2959.

[16] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[17] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, , L. Kotthoff, , and J. Vanschoren, Eds. Springer International Publishing, 2019, pp. 63–77.

[18] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer International Publishing, 2019.

[19] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[20] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 3 2008.

[21] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 6 2002.

[22] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2011.

[23] D. Izzo, M. Ruciński, and F. Biscani, "The generalized Island model," *Studies in Computational Intelligence*, vol. 415, no. January 2012, pp. 151–169, 2012.

[24] L. Hill and M. Flack, "The Physiological Influence of Ozone," *Proceedings of the Royal Society B: Biological Sciences*, vol. 84, no. 573, pp. 404–415, 12 1911.

[25] European Commission, "Directive 2002/3/EC of the European Parliament and of the council of 12 February 2002 relating to ozone in ambient air," *Official Journal of the European Union*, 2002.

[26] J. Hooyberghs, C. Mensink, G. Dumont, F. Fierens, and O. Brasseur, "A neural network forecast for daily average PM10 concentrations in Belgium," *Atmospheric Environment*, vol. 39, no. 18, pp. 3279–3289, 6 2005.

[27] O. M. Agudelo, O. B. Mendoza, P. Viaene, and B. De Moor, "Assimilation of ozone measurements in the air quality model AURORA by using the Ensemble Kalman Filter," *Proceedings of the IEEE Conference on Decision and Control*, no. ii, pp. 4430–4435, 2011.

[28] O. M. Agudelo, P. Viaene, and B. De Moor, "Improving the PM10 estimates of the air quality model AURORA by using Optimal Interpolation," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1154–1159, 2015.

[29] R. Langone, O. M. Agudelo, B. De Moor, and J. A. Suykens, "Incremental kernel spectral clustering for online learning of non-stationary data," *Neurocomputing*, vol. 139, pp. 246–260, 2014.

[30] B. S. Freeman, G. Taylor, B. Gharabaghi, and J. Thé, "Forecasting air quality time series using deep learning," *Journal of the Air & Waste Management Association*, vol. 68, no. 8, pp. 866–886, 8 2018.

[31] O. F. Cook, "FACTORS OF SPECIES-FORMATION." *Science (New York, N.Y.)*, vol. 23, no. 587, pp. 506–7, 3 1906.

[32] M. Tomassini, *Spatially Structured Evolutionary Algorithms*. Springer, 2005.

[33] W. N. Martin, J. Lienig, and J. P. Cohoon, "C6.3 Island (migration) models : Evolutionary algorithms based on punctuated equilibria," *Handbook of Evolutionary Computation*, 1997.

[34] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[35] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 12 1997.

[36] D. Kraft, "A software package for sequential quadratic programming, Technical Report DFVLR-FB 88-28," Braunschweig, p. 33, 1988.

[37] S. G. Nash, "Newton-Type Minimization Via the Lanczos Method," *SIAM Journal on Numerical Analysis*, vol. 21, no. 4, pp. 770–788, 8 1984.

[38] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Society for Industrial and Applied Mathematics, 1 2000.

[39] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, G. Brain, I. Osdi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," in *12th USENIX conference on Operating Systems Design and Implementation*, 2015.

[40] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2 2015, pp. 448–456.

[41] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 6 2017, pp. 972–981.

[42] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 12 1943.

[43] F. Chollet and others, "Keras," 2015. [Online]. Available: https://keras.io

[44] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, San Diego, 12 2015.

[45] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *30th International Conference on Machine Learning, ICML 2013*, no. PART 3, Atlanta, Georgia, USA, 2013, pp. 1139–1147.
[46] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
[47] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *COLT 2010 - The 23rd Conference on Learning Theory*, vol. 12, 2010, pp. 257–269.
[48] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *CoRR*, 12 2012.
[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
[50] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in Neural Information Processing Systems*, pp. 1027–1035, 2016.
[51] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–16, 2019.
[52] Ian Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[53] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas, "Blue Gene/L torus interconnection network," *IBM Journal of Research and Development*, vol. 49, no. 2-3, pp. 265–276, 3 2005.
[54] T. Bäck and F. Hoffmeister, "Extended Selection Mechanisms in Genetic Algorithms," in *Proceedings of the 4th International Conference on Genetic Algorithms*, 1991, pp. 92–99.
[55] D. Whitley and J. Kauth, "GENITOR: A different genetic algorithm," in *Proceedings of the Rocky Mountain conference on artificial intelligence, 1988*, Fort Collins, Colorado, 1988, pp. 88–101.
[56] T. D. Price, A. Qvarnström, and D. E. Irwin, "The role of phenotypic plasticity in driving genetic evolution," *Proceedings of the Royal Society B: Biological Sciences*, vol. 270, no. 1523, pp. 1433–1440, 7 2003.
[57] D. J. Whitehead, C. O. Wilke, D. Vernazobres, and E. Bornberg-Bauer, "The look-ahead effect of phenotypic mutations," *Biology Direct*, vol. 3, p. 18, 5 2008.
[58] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 10 2006.
[59] C. Bergmeir, R. J. Hyndman, and B. Koo, "A note on the validity of cross-validation for evaluating autoregressive time series prediction," *Computational Statistics and Data Analysis*, vol. 120, pp. 70–83, 4 2018.
[60] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: WORLD SCIENTIFIC, 11 2002.
[61] S. C. Cunnane, *Survival of the fattest: The key to human brain evolution*. World Scientific Publishing Co., 1 2005.
[62] T. G. Mattson, B. A. Sanders, and B. Massingill, *Patterns for parallel programming*. Addison-Wesley, 2005.
[63] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, "Parallel distributed computing using Python," *Advances in Water Resources*, vol. 34, no. 9, pp. 1124–1139, 2011.
[64] G. Hohpe and B. Woolf, *Enterprise integration patterns : designing, building, and deploying messaging solutions*. Addison-Wesley, 2004.
[65] "RabbitMQ tutorial - Work Queues — RabbitMQ." [Online]. Available: https://www.rabbitmq.com/tutorials/tutorial-two-python.html
[66] "VLAAMS SUPERCOMPUTER CENTRUM ANNUAL REPORT 2018," 2018. [Online]. Available: https://www.vscentrum.be/



Konstantinos Theodorakos was born in Chalkida, Greece, on 12 December 1984. He has a Bachelor of Science: Informatics Engineering (2013) from the Technological Educational Institute (TEI) of Crete (Technical College) Greece, with the thesis: "Client-Server Online Storage Platform with the usage of C# .NET Framework and TCP/IP socket programming". He received a Master of Computer Science: Software Engineering (2017) from the Universiteit Antwerpen in Belgium, with the thesis: "Modelling of Stochastic Compartmental Spatio-Temporal Epidemic Simulations with Cellular Automata and acceleration with CPU and GPGPU parallelism". He has worked as an Electronic Hardware Technician (2008-2011) and as a Numerics Software Developer for Spatial and Spatio-Temporal Machine Learning on Geographic Information Systems (2013-2018). He is currently a Ph.D. student in Engineering Science, at the Department of Electrical Engineering (ESAT) and the STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics of the KU Leuven in Belgium. He is working on time series analysis and modelling in data science applications.



Oscar Mauricio Agudelo received the B.S. degree in electronics engineering from the Universidad Autónoma de Occidente, Cali, Colombia, in 1997, the M.S. degree in industrial control engineering from the Universidad de Ibagué (in cooperation with KU Leuven and Universiteit Gent), Ibagué, Colombia, in 2004, and the Ph.D. degree in electrical engineering from KU Leuven, Leuven, Belgium, in 2009. From 1997 to 2004, he worked at the Universidad Autónoma de Occidente as a full time professor of control and automation. After his Ph.D., he held a postdoctoral position and later on a research manager position in the research group STADIUS at the Department of Electrical Engineering of KU Leuven. He is currently a project coordinator on systems and control in the same research group. His research interests are in model reduction techniques, systems and control theory, machine learning, model predictive control, data assimilation, deep learning, polynomial optimization, system identification, and analysis and design of intelligent control systems.



Joachim Schreurs was born in Hasselt Belgium, October 13, 1994. In 2015 he received a Bachelor's degree in Engineering Science, Electrical Engineering and Computer Science and in 2017 a Master's degree in Engineering Science, Mathematical Engineering, both at the KU Leuven. He is currently a doctoral student in machine learning, at the STADIUS research division of the Department of Electrical Engineering (ESAT) at KU Leuven, under the supervision of prof. Johan A. K. Suykens. Joachim's scientific interests include machine learning, kernel methods, generative models, robust statistics and sampling algorithms for kernel methods.



Johan A.K. Suykens was born in Willebroek Belgium, May 18 1966. He received the master degree in Electro-Mechanical Engineering and the PhD degree in Applied Sciences from the Katholieke Universiteit Leuven, in 1989 and 1995, respectively. In 1996 he has been a Visiting Postdoctoral Researcher at the University of California, Berkeley. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a full Professor with KU Leuven. He is author of the books "Artificial Neural Networks for Modelling and Control of Non-linear Systems" (Kluwer Academic Publishers) and "Least Squares Support Vector Machines" (World Scientific), co-author of the book "Cellular Neural Networks, Multi-Scroll Chaos and Synchronization" (World Scientific) and editor of the books "Nonlinear Modeling: Advanced Black-Box Techniques" (Kluwer Academic Publishers), "Advances in Learning Theory: Methods, Models and Applications" (IOS Press) and "Regularization, Optimization, Kernels, and Support Vector Machines" (Chapman & Hall/CRC). In 1998 he organized an International Workshop on Nonlinear Modelling with Time-series Prediction Competition. He has served as associate editor for the IEEE Transactions on Circuits and Systems (1997-1999 and 2004-2007), the IEEE Transactions on Neural Networks (1998-2009) and the IEEE Transactions on Neural Networks and Learning Systems (from 2017). He received an IEEE Signal Processing Society 1999 Best Paper Award, a 2019 Entropy Best Paper Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He has served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven 2002), as a program co-chair for the International Joint Conference on Neural Networks 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007, a co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning, and chair of ROKS 2013. He has been awarded an ERC Advanced Grant 2011 and 2017, and has been elevated IEEE Fellow 2015 for developing least squares support vector machines. He is currently serving as program director of Master AI at KU Leuven.



Bart De Moor received the doctoral degree in applied sciences, in 1988, from the Katholieke Universiteit, Leuven, Belgium. He was a visiting research associate from 1988 to 1989 in the Department of Computer Science and Electrical Engineering, Stanford University, Stanford, CA. He is currently a full professor at the Katholieke Universiteit, Leuven. His research interests include numerical linear algebra, system identification, advanced process control, data mining, and bio-informatics. He is the (co-)author of several books and several hundreds of papers, some of which have been awarded. He received the Leybold-Heraeus Prize in 1986, the Leslie Fox Prize in 1989, the Guillemin-Cauer Best Paper Award, of the IEEE Transactions on Circuits and Systems, in 1990, the biannual Siemens prize in 1994, and became a Laureate of the Belgian Royal Academy of Sciences in 1992.